Closed-loop optical stimulation & recording system with GPU-based real-time spike sorting

Ling Wang[*a,b], Thoa Nguyen[a,b], Henrique Cabral[b,c], Barbara Gysbrechts[a,b], Francesco Battaglia[b,c], Carmen Bartic[a,b]

[a]Department of Physics and Astronomy, KU Leuven, Celestijnenlaan 200D, 3001 Heverlee, Belgium Belgium; [b]NERF- Neuro-Electronics Research Flanders, Kapeldreef 75, 3001 Leuven, Belgium; [c]Donders Institute, Radboud University Nijmegen, Kapittelweg 29, 6525 EN Nijmegen, Netherlands

ABSTRACT

Closed-loop brain computer interfaces are rapidly progressing due to their applications in fundamental neuroscience and prosthetics. For optogenetic experiments, the integration of optical stimulation and electrophysiological recordings is emerging as an imperative engineering research topic. Optical stimulation does not only bring the advantage of cell-type selectivity, but also provides an alternative solution to the electrical stimulation-induced artifacts, a challenge in closed-loop architectures. A closed-loop system must identify the neuronal signals in real-time such that a strategy is selected immediately (within a few milliseconds) for delivering stimulation patterns. Real-time spike sorting poses important challenges especially when a large number of recording channels are involved. Here we present a prototype allowing simultaneous optical stimulation and electro-physiological recordings in a closed-loop manner. The prototype was implemented with online spike detection and classification capabilities for selective cell stimulation. Real-time spike sorting was achieved by computations with a high speed, low cost graphic processing unit (GPU). We have successfully demonstrated the closed-loop operation, i.e. optical stimulation in vivo based on spike detection from 8 tetrodes (32 channels). The performance of GPU computation in spike sorting for different channel numbers and signal lengths was also investigated.

Keywords: Optogenetics, spike sorting, GPU processing, optical stimulation, closed loop; multiple channels, template matching

1. INTRODUCTION

Through electrophysiological signals, neurons communicate each other for diverse purposes. The signals generated by individual neurons are referred as spikes that can be detected by microelectrodes implanted in the living brain tissues. The signal recorded by such a microelectrode depends on the electrode size and position with respect to the firing source neurons Neuronal spike sorting is the process of grouping spikes into clusters based on the similarity of their spike patterns recorded by neighboring electrodes in order to determine neuron source of each spike. Therefore, detection and sorting of neuronal spikes offer a valuable approach to understand the function of neural circuits

Recently, the rapid growth of brain-computer interface architectures and their wide application potential have induced extensive interest in the combining neural stimulation with recording capabilities to perform either the open-loop or the closed-loop operation[1]. Closed-loop operation can provide on-line stimulation strategies, flexibly depending on the real-time activity of neurons under study. A closed-loop system is becoming more interesting in the optogenetic experiments, since the cell-type selectivity of optical stimulation and thefast response speed of opsins allow stimulation on multiple cell populations with a high target and temporal precision[2, 3].

However, for real-time applications, closed-loop systems suffer from the low speed of the spike sorting processes, especially for the experiments with multiple signal detection channels. Therefore, spike sorting is usually avoided in the closed-loop experiments being replaced by spike detection (i.e. detecting the presence of action potentials only, for example, by applying a voltage threshold). This can be relatively easy to be implemented by using a central processing unit (CPU) of a compouter, or even in a hardware[4]. However, this method can't identify source activity changes in the closed-loop operation[1]. A graphics processing unit (GPU) with hundreds of processing cores provides highly parallel

[*] E-mail: Ling.Wang@biw.kuleuven.be

Biophotonics: Photonic Solutions for Better Health Care IV, edited by Jürgen Popp, Valery V. Tuchin, Dennis L. Matthews, Francesco S. Pavone, Proc. of SPIE Vol. 9129, 91293U · © 2014 SPIE · CCC code: 0277-786X/14/$18 · doi: 10.1117/12.2052098

computational capability, thereby accelerating numerical signal processing in a real-time system[5-8]. In the multi-channel signal recording system, individual channels (or tetrodes) can be treated independently, which leads to a well parallelizable problem. In this study, we developed a system allowing simultaneous optical stimulation and 32- channel electro-physiological recordings in a closed-loop manner. Real-time spike sorting was achieved by computations with a high speed, low cost GPU card.

2. MATERIALS AND METHODS

2.1 Closed-loop system architecture

The overall hardware system consists of three main parts: (1) a headstage that amplifies, multiplexes and digitally transmits the recorded signals to the digital I/O board, (2) a LED driver that controls the brightness of the LED through the current pulses, and (3) a data acquisition card (DAQ) that provides digital I/O and analog output pulses. The details of the microsystem design and micro-electrode drive were described in our other publications[9, 10]. Briefly, the optic fibers and the eight tetrodes connected to the implanted headstagevia a Neuroalynx electrode interface and micro-electrode drive. Each tetrode consisted of a twisted bundle of four microwires and was coupled into a guiding unit which could be vertically moved by a micro screw. The recorded 32 channel neural signals were first amplified and filtered by the amplifier board armed with an Intan Technology RHA2132 chip. The analog signals were digitalized with a sampling rate of 12.5kHz and multiplexed on board before they were fed into the DAQ card (National Instrument, PCI-6259M, TX, USA). Digital isolators (ADUM1400 series) were placed to decouple the ground of the amplifier board from the noisy computer ground to further suppress the effect of external noise. The amplifier board operated on two battery-supplied power sources: a regulated +3V and +5V source. A 2.5V reference voltage is used for the ADC conversion. A simple high-power LED driver supporting pulse-width modulation (PWM) is built to control the emission power of a 470 nm wavelength LED (Thorlabs, M470F1) for optical stimulation.

Custom software was developed to control the whole system, including system synchronization, data acquisition, result visualization and online signal processing. We adopted LabVIEW (Version 2012, 64bit, National Instrument, TX, USA) as the platform to implement the program that contains two concurrent threads, one for data acquisition from the headstage and storage, and the other for signal processing (spike sorting). The program automatically assigned CPU cores to the threads, and the data were shared via global variables which were updated in real-time. With this strategy, the signal processing thread always preceded the latest data, but could suffer a data loss risk when it spent more time than the data acquisition. Therefore, a computation speed of spike sorting faster than data acquisition speed is required. In our system, neural signals were recorded from tetrodes every 8 ms, corresponding to 100 sampling points/channel. It means that the signal processing must be finished within 8 ms.

2.2 Signal processing on GPU

The signal processing algorithm consisted of steps of data conversion, de-multiplexing, filtering, spike detection and classification. The raw binary digital data were firstly converted to analog voltages in 16 bit integer type. Then, the date were transferred from the host computer memory to the GPU memory and converted into 32-bit floating-point type (single precision). The GPU de-multiplexed the signals to separate them into their constituent input channels and re-scaled them to real voltages. A second-order elliptic zero-phase bandpass filter was applied on the signals before spike detection. Spikes were detected by applying a negative amplitude threshold that was calculated based on the standard deviation of the background noise using the formula proposed by Quiroga *et al*[11]. We assumed that there were no more than 5 spikes for each channel within 8 ms recording period. Spike classification was carried out using a multi-channel template matching process. Templates were extracted from a baseline recording period at the beginning of the session. Our current template definition is based on a standard mixture-of-Gaussians expectation maximization (EM) algorithm[12]. Online spike sorting was then performed using the simplest form of template matching, i.e. using the correlation between single spike waveforms and each template. The normalized correlation of the waveform with all templates was taken, and the spike was assigned to the template resulting in the maximum score of the correlation. With the convolution theorem, a fast Fourier transformation algorithm was used to implement the correlation instead of the convolution in order to reduce the computation time. Finally, the results of the spike sorting were sent back to the host memory. The system generated a stimulation strategy immediately and gave an order to the LED. Figure 1 shows the architecture of the closed-loop system and the data flow during the experiment.

Both programming and computation in this work were implemented with a workstation hosting an NVIDIA GeForce GTX 680 GPU (PCIe 3.0 16× interface, 1536 cores at 1.006 GHz, 2.0 GB graphics memory), an Intel i7-3770 CPU (3.9 GHz) and a 16 GB host memory. The operating system was Microsoft Windows 7 Professional. In contrast to programming codes directly through NVIDIA's Compute Unified Device Architecture (CUDA), we employed the commercial software library, ArrayFire (Version 2.0, AccelerEyes, GA, USA) that makes GPU programming simple by packaging most basic CUDA codes into array-based functions. This relatively simple scheme for GPU programming and GPU-memory access allows for faster and more flexible adaptation of the existing signal processing routines. Since our system adopted Labview as a coding tool for data acquisition and system control, while the GPU-based spike sorting codes were performed in VC++ (Visual Studio 2008, Professional Edition), we proposed a strategy that employed a shared library compiled from VC++ scripts to implement the spike sorting. The VC++ scripts were first compiled to generate a dynamic link library (DLL) which was available for other program platforms such as LabView in this project.

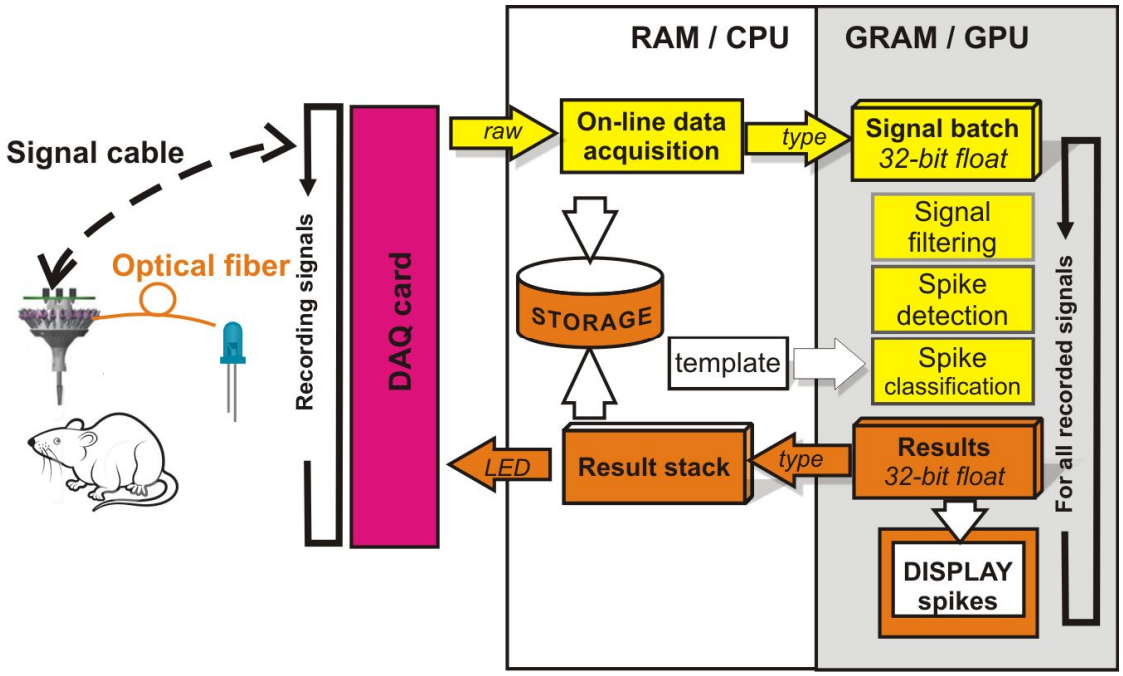

Figure 1 Architecture of the closed-loop system and the data flow.

2.3 Animal surgery

To prove the system's functionality in awake animals, microwire array recordings were taken from the dorsal hippocampus of adult male Sprague-Dawley rats weighing 350 g (Charles River Laboratories; Wilmington, MA, USA). Animal work was conducted in accordance with protocols approved by the KULeuven animal ethics committee and in accordance with the European Communities Council Directive of November 24, 1986 (86/609/EEC). Rats were anesthetized with 1.5–3.0% inhaled isoflurane and given a subcutaneous injection of buprenorphine (0.05 mg/kg) to minimize pain. A craniotomy was made over the right dorsal hippocampus, centered at 3.5 mm posterior and 2.8 lateral to bregma. The dura was incised with a sterile syringe needle and a microwire array (Tucker Davis Technologies) with 22 tetrodes was implanted. The craniotomy was sealed with a bio-compatible, silicon based elastomer (Kwil-Silfrom World Precision Instruments), skull screws (Amazon Supply, US) were implanted, and UV-curable cement (SDI, AUS) was applied to secure the array. The rats were returned to their normal housing, rest and recovered post-operatively for 5-8 days before the recordings begin. The use of animals and the procedures were approved by the Ethical Committee for Animal Welfare (ECD, Ethische commissie Dierenwelzijn) of the KULeuven.

RESULTS AND DISCUSSION

Validation of the hardware system and the spike sorting algorithm have been described and discussed in detail in [9, 10]. In this paper, we report the results of the GPU computation and discuss its performance in different experiment conditions.

Table 1 lists the average GPU computation times for processing a signal batch of 32 channels with data lengths of 100 and 200 sampling points. All the processing times in this paper were averaged from tests with 1000 continuous real experiment signal recordings. For the signal sampling rate of 12.5 kHz in our system, data lengths of 100 and 200 sampling points corresponds to 8 ms and 16 ms recording periods, respectively. When the sampling rate is doubled to 25 kHz, 200 sampling points corresponds to 8 ms. One can see that the average total processing times of both data lengths are 2.839 ms and 3.539 ms, respectively. It means that most stimulation orders can be sent to the light sources within 4 ms once a signal recoding has been finished. Figure 2 illustrates the percentages of the GPU computation times for each step presented in Table 1. For both data lengths, signal filtering is the main consumer of the GPU computational resource. Therefore, a faster digital filtering algorithm or even a filtering hardware is strongly desired, especially when data length is long. In many applications, data transfer between the computer and GPU memory is significant extra time consumption when large data communication is required. For spike sorting in this study, copying data from the host computer to the GPU device takes ~ 10% of the total time, while the inverse process takes a very small portion. This can be attributed to the relatively small data flow in this study, and the improvement of the data transfer technique for the GPU devices.

Table 1 GPU computation times for processing a signal batch of 32 channels with 100 and 200 sampling points.

Sampling points	100	200
Processing	Time(ms)	
Memory copy (host to device)	0.319	0.336
Signal filtering	0.961	1.533
Spike detection	0.625	0.726
Template matching	0.889	0.898
Memory copy (device to host)	0.045	0.046
Total	2.839	3.539

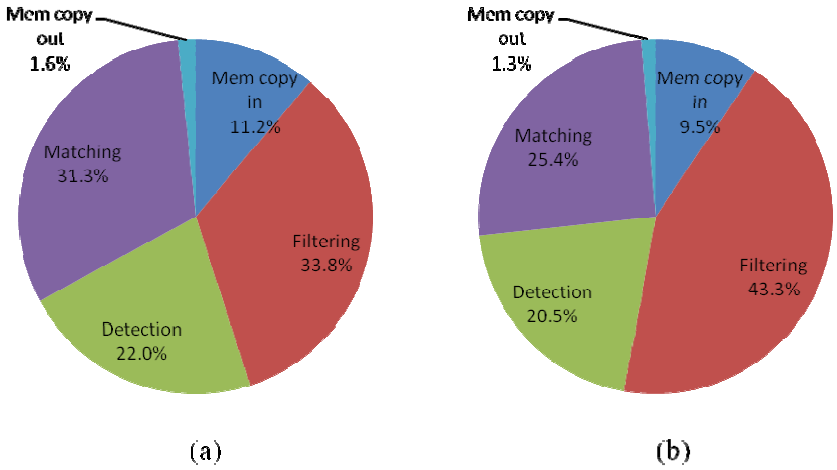

Figure 2 Percentage (mean of 1000 tests) of the GPU computation times for processing a signal batch of 32 channels with (a) 100 and (b) 200 sampling points. Mem copy in: data copy from host to device memory; Filtering: Signal filtering; Detection: spike detection; Matching: template matching; Mem copy out: data copy from device to host memory.

Figure 3 shows the total GPU computation times for the signal batches with different data lengths and channel numbers. The white line is the boundary between the regions of computation time > 8 ms and < 8ms. The computation time in the upper left corner region is shorter than 8ms, while in the lower right corner is longer than 8ms. One can see from this figure that the computation time increases with the channel number and data length. The spike sorting can be finished within 8 ms for more than 500 recording channels when the data length is 100 and 200 sampling points. Within 8 ms, data length of up to ~900 sampling points for 32 channels can be processed. Therefore, our system has the potential to extend the sampling rate of more than 100 kHz/channel, which could help to increase the SNR of the signal measurement.

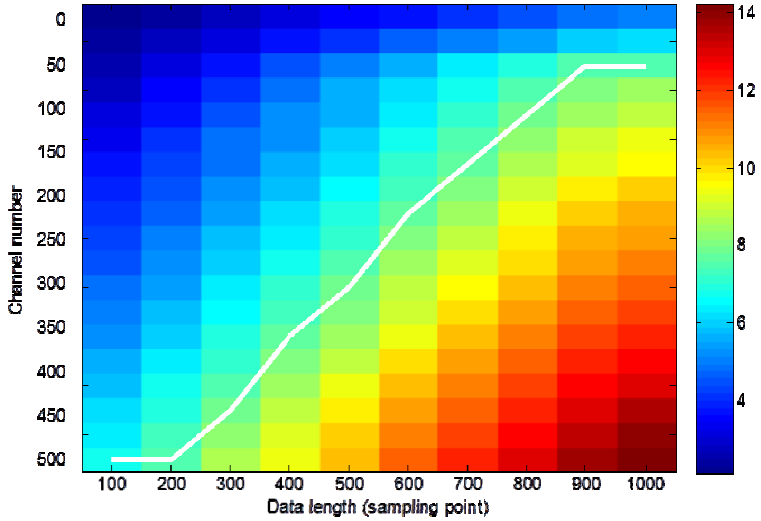

Figure 3 The total GPU computation time (unit: ms) versus data length and channel number. The location of the white line is at a computation time of 8 ms.

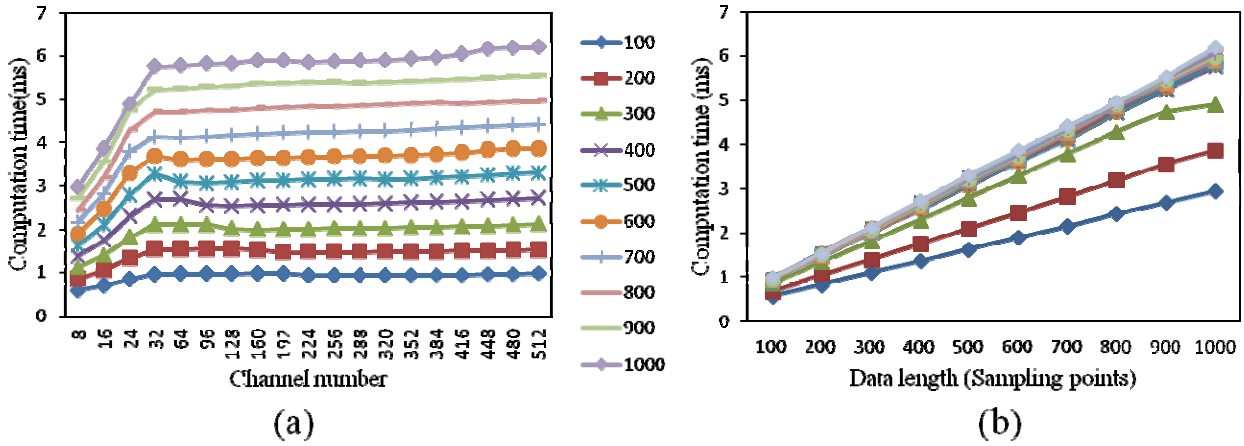

Figure 4 The GPU Computation times for signal filtering versus (a) channel number and (b) data length.

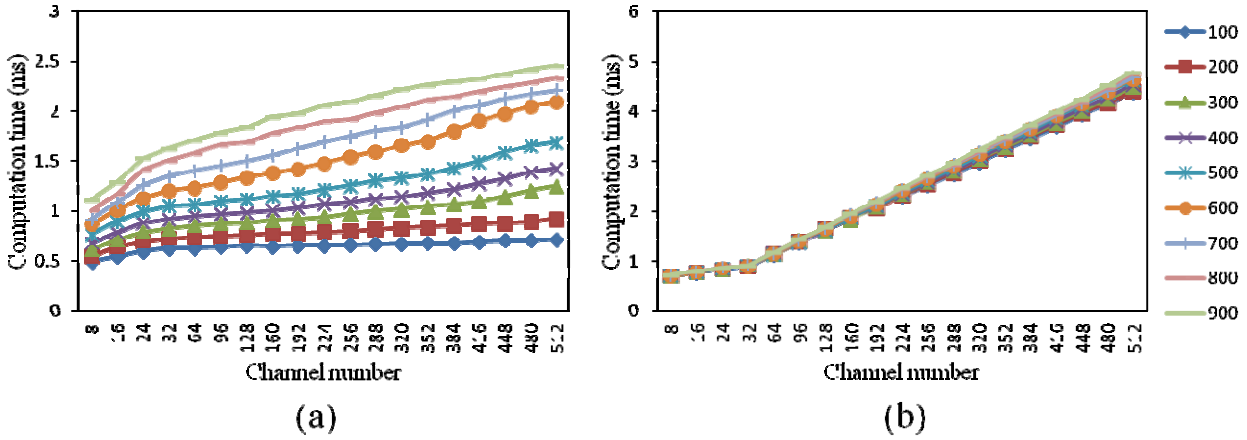

Figure 5 The GPU Computation times versus channel number for (a) spike detection and (b) template matching.

Figure 4 illustrates the computation times for signal filtering. In Figure 4 (a), the computation times increases with the channel number when it is less than 32 channels, while interestingly they keep almost constant after that. This happens on all the data lengths ranging from 100 to 1000 sampling points. It tells us that the computation of the filtering algorithm used in our project is highly parallel and each core of the GPU card runs almost independently for each signal channel, demonstrating the significant advantage of the GPU. In Figure 4 (b), the computation times have a linear relationship with the data length.

Figure 5 (a) and (b) plot the computation times for the spike detection and template matching. In both plots, the computation time increases with the channel number. However, the computation time in Figure 5 (b) increases faster than in (a), and stays almost the same for any data length. The latter is easy to be understood, since the spikes have been cropped from the signals for template matching. Yet, the linear relationship between the computation time and the channel number explores a fact that the computation times linearly depend on the number of the spikes as is out of our expectation.

CONCLUSION

We have presented a closed-loop optical stimulation and recording system with GPU-based real-time spike sorting. The average processing time for a data set of 32 channels and 8 ms recording period (12.5 kHz sampling rate) was 2.839 ms which was sufficiently short to decide a stimulation strategy. The performance of GPU computation in spike sorting for different channel numbers and signal lengths shows that GPU computation has more potential in spike sorting for the large number of signal.

ACKNOWLEDGMENTS

This work has been supported by the FP7 EC project ENLIGHTENMENT that acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission.

REFERENCES

[1] Franke, F., D. Jackel, J. Dragas, J. Muller, M. Radivojevic, D. Bakkum, and A. Hierlemann, “High-density microelectrode array recordings and real-time spike sorting for closed-loop experiments: an emerging technology to study neural plasticity,” Front Neural Circuits, 6, 105 (2012).

[2] Zhang, Y. P., and T. G. Oertner, “Optical induction of synaptic plasticity using a light-sensitive channel,” Nat Methods, 4(2), 139-41 (2007).

[3] Henderson, J. M., T. Federici, and N. Boulis, "Optogenetic neuromodulation," Neurosurgery, 64(5), 796-804; discussion 804 (2009).
[4] Guillory, K. S., and R. A. Normann, "A 100-channel system for real time detection and storage of extracellular spike waveforms," J Neurosci Methods, 91(1-2), 21-9 (1999).
[5] Chen, Y. Y., Y. M. Tsai, and L. G. Chen, [Algorithm and implementation of multi-channel spike sorting using GPU in a home-care surveillance system] IEEE, Barcelona, Spain(2011).
[6] Courtecuisse, H., H. Jung, J. Allard, C. Duriez, D. Y. Lee, and S. Cotin, "GPU-based real-time soft tissue deformation with cutting and haptic feedback," Prog Biophys Mol Biol, 103(2-3), 159-68 (2010).
[7] Hafeez, A., W. Asghar, M. M. Rafique, S. M. Iqbal, and A. R. Butt, "GPU-based real-time detection and analysis of biological targets using solid-state nanopores," Med Biol Eng Comput, 50(6), 605-15 (2012).
[8] Wang, L., B. Hofer, J. A. Guggenheim, and B. Povazay, "Graphics processing unit-based dispersion encoded full-range frequency-domain optical coherence tomography," J Biomed Opt, 17(7), 077007 (2012).
[9] Nguyen, T. K. T., [Microsystem design and signal processing towards closed-loop bi-directional neural interfaces] Katholieke Universiteit Leuven, (2013).
[10] Nguyen, T. K. T., Z. Navratilova, H. Cabral, L. Wang, G. Gielen, F. P. Battaglia, and C. Bartic, "A 32-channel low-noise recording system with online spike sorting," Journal of Neural Engineering, (In revision).
[11] Quiroga, R. Q., Z. Nadasdy, and Y. Ben-Shaul, "No Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," Neural Computation, 16(8), 1661-1687 (2004).
[12] Harris, K. D., D. A. Henze, J. Csicsvari, H. Hirase, and G. Buzsaki, "Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements," J Neurophysiol, 84(1), 401-14 (2000).